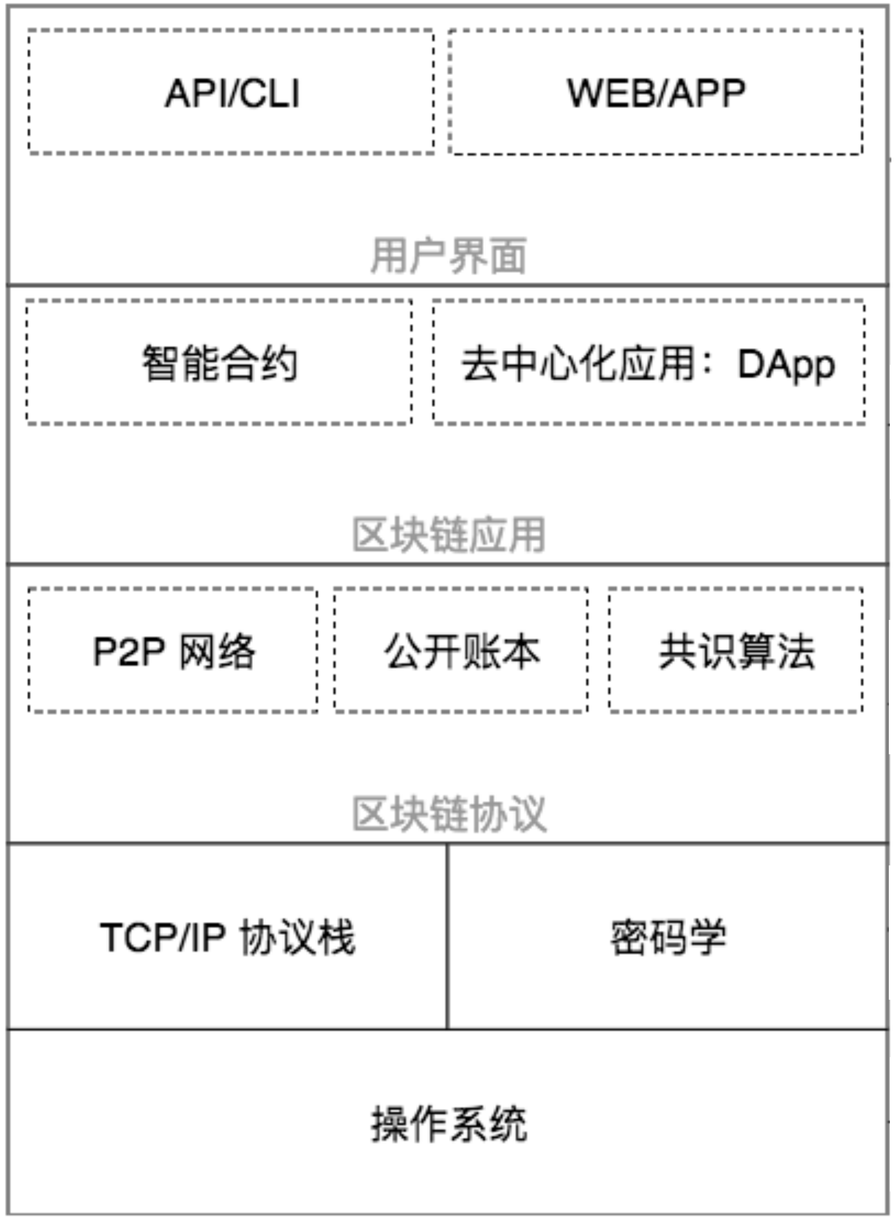

掘金
IT大咖说
知识共享平台
以太坊智能合约 + DApp 从入门到上线
王仕军

BIG PICTURE


API/CLI
WEB/APP
用户界面
智能合约
去中心化应用：DApp
区块链应用
P2P 网络
公开账本
共识算法
区块链协议
TCP/IP 协议栈
密码学
操作系统

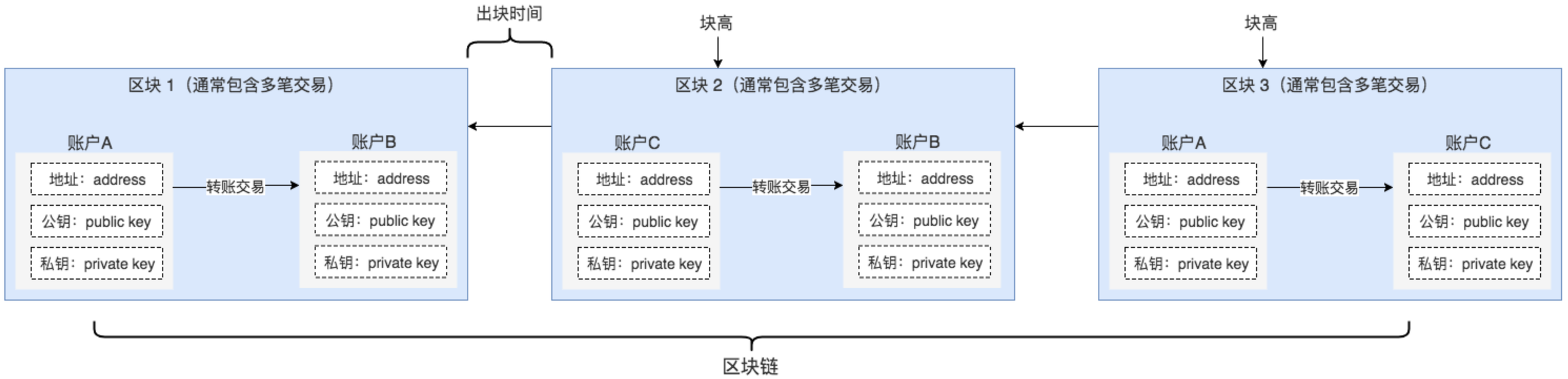
主要议题

- 最少必要知识
- 开发环境准备
- 智能合约工作流
- DApp 构建和上线

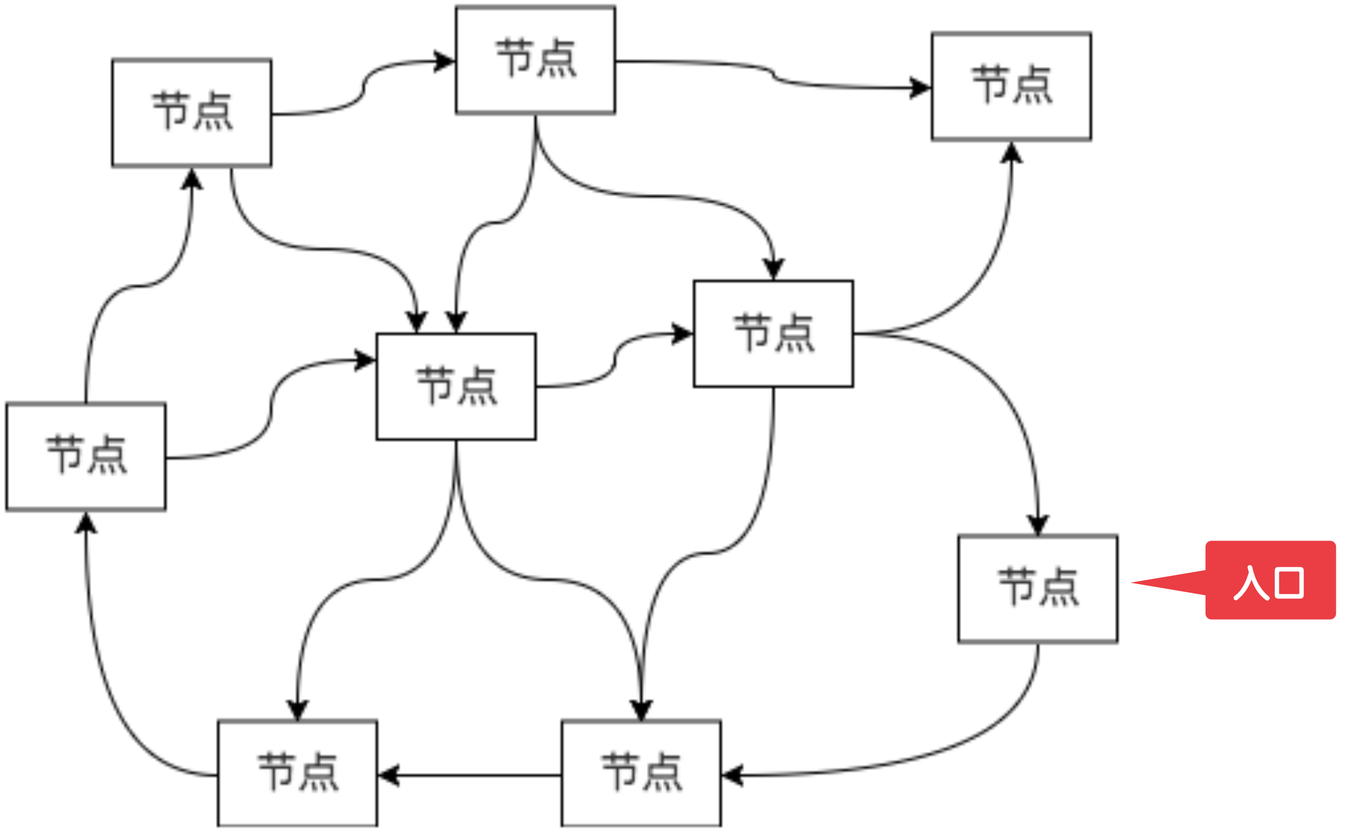
最小必要知识

- 账户、交易、区块、区块链、块高、区块时间
- 以太坊网络、智能合约、DApp

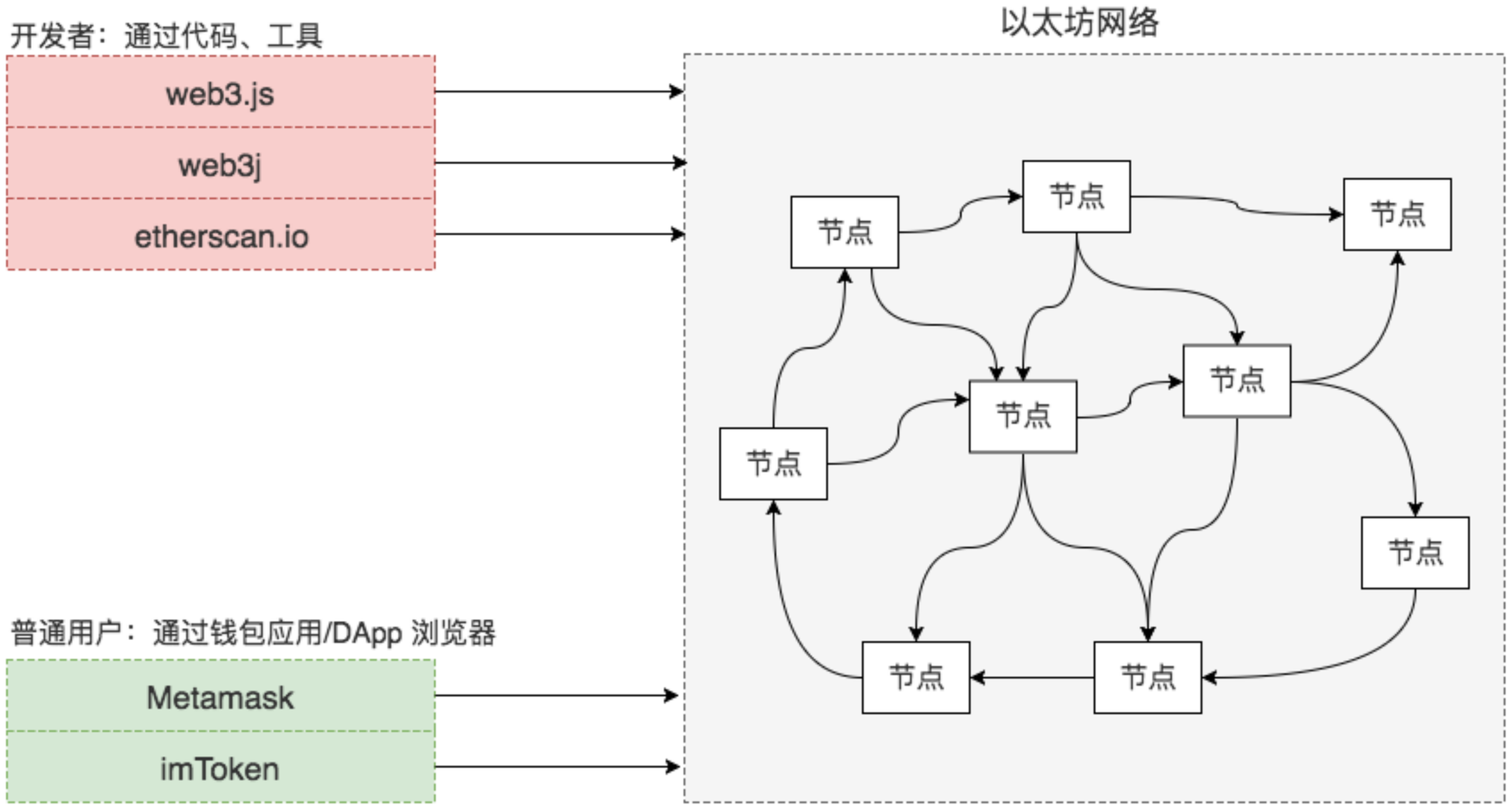
一图胜千言


出块时间
块高
块高
区块 1（通常包含多笔交易）
区块 2（通常包含多笔交易）
区块 3（通常包含多笔交易）
账户A
地址：address
公钥：public key
私钥：private key
转账交易
账户B
地址：address
公钥：public key
私钥：private key
账户C
地址：address
公钥：public key
私钥：private key
转账交易
账户B
地址：address
公钥：public key
私钥：private key
账户A
地址：address
公钥：public key
私钥：private key
转账交易
账户C
地址：address
公钥：public key
私钥：private key
区块链

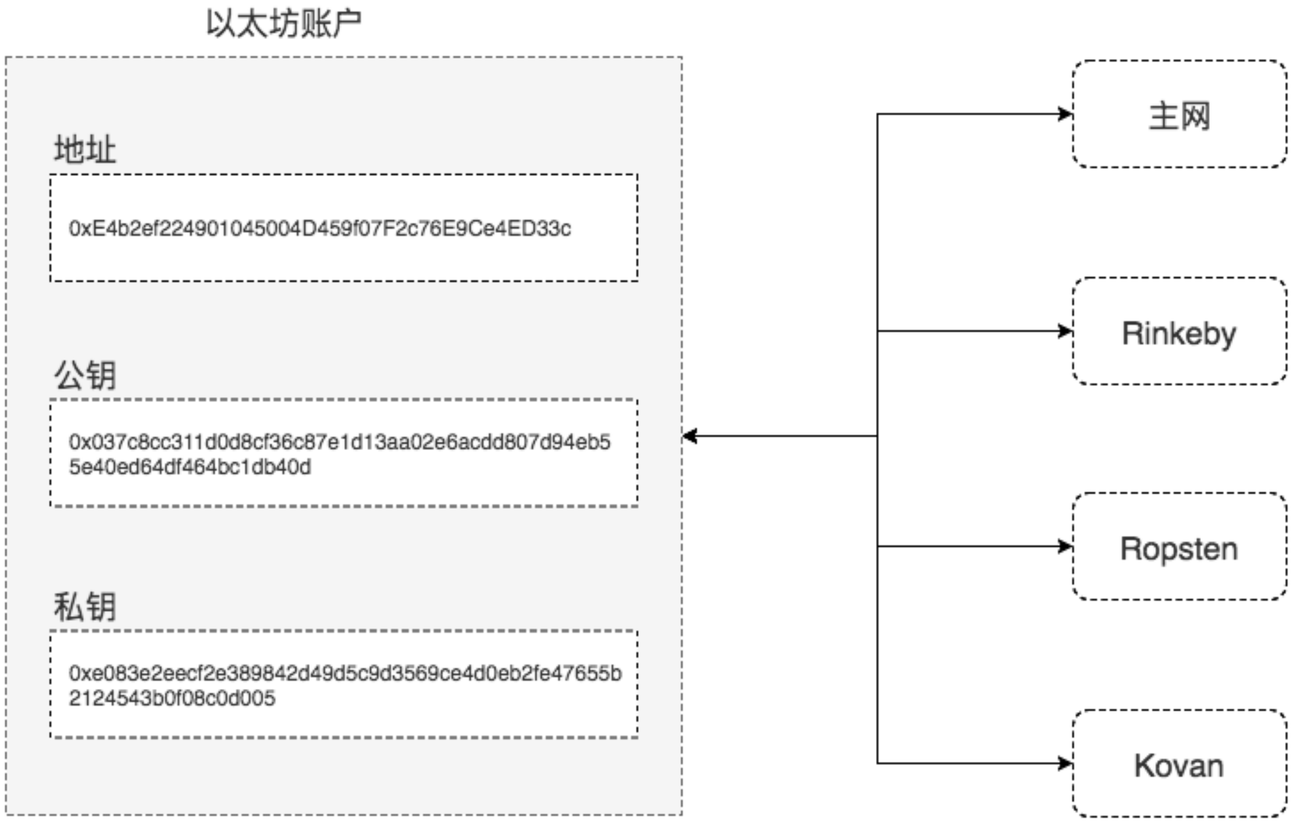
以太坊网络 = 分布式网络



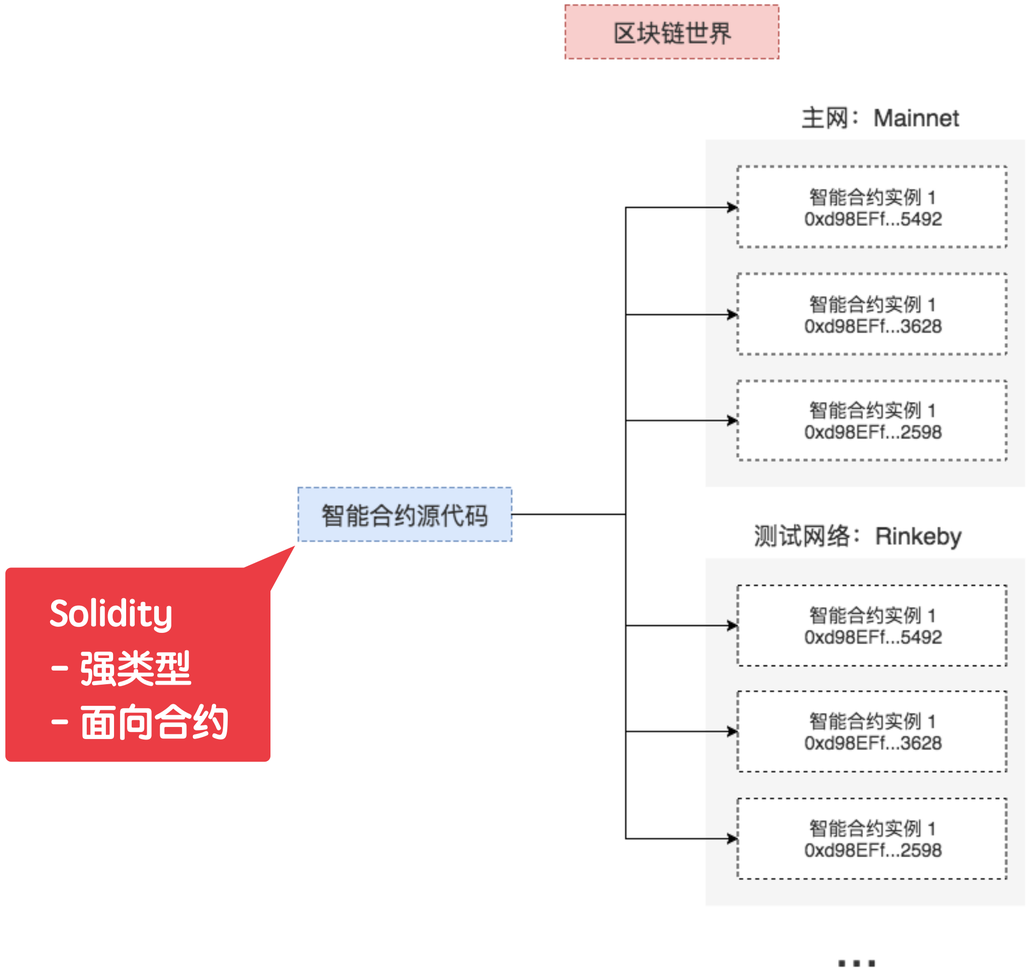
如何与以太坊网络交互


开发者：通过代码、工具
web3.js
web3j
etherscan.io
以太坊网络
节点
节点
节点
节点
节点
节点
节点
节点
节点
普通用户：通过钱包应用/DApp 浏览器
Metamask
imToken

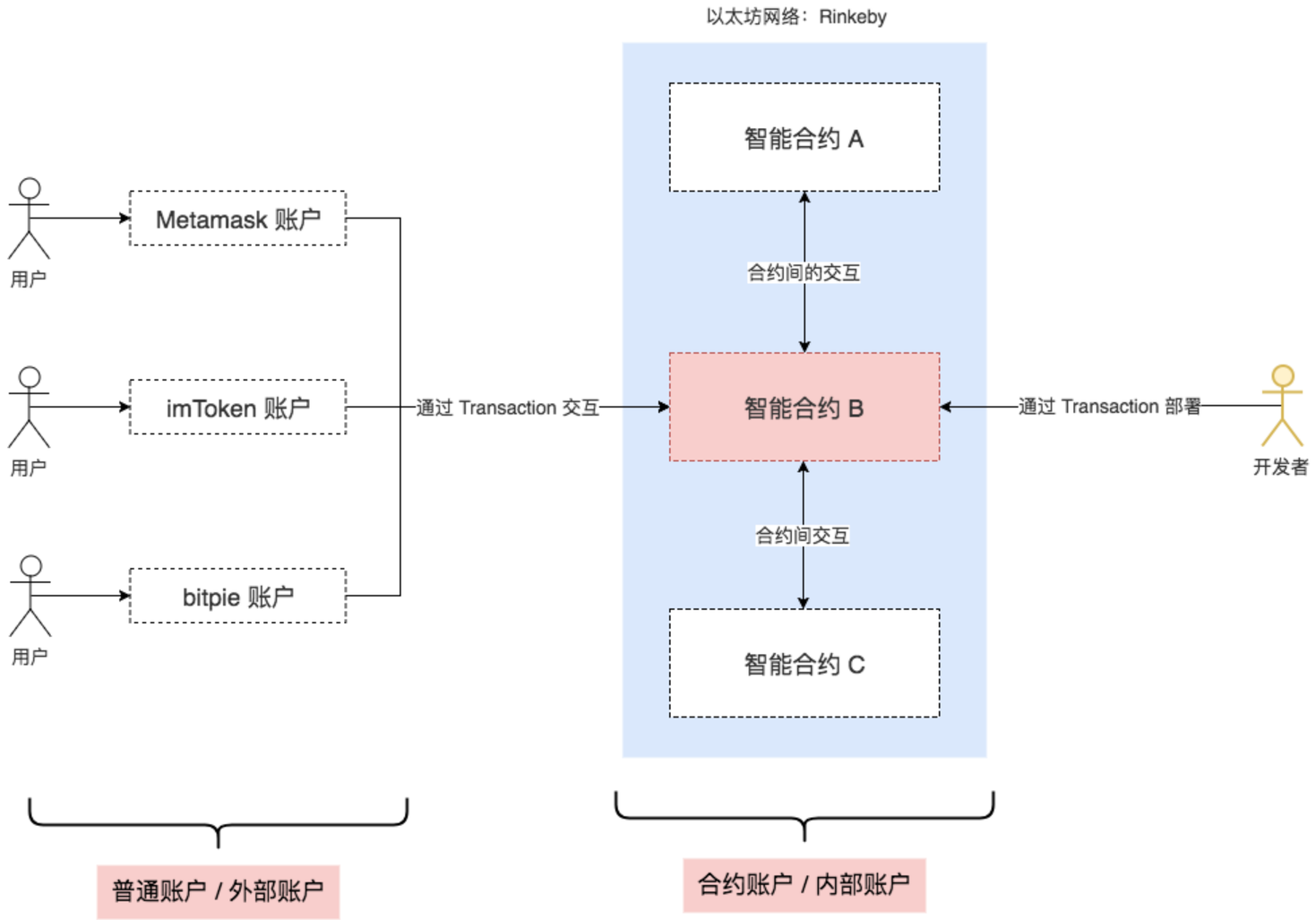
以太坊账户是啥?


以太坊账户
地址
0xE4b2ef224901045004D459f07F2c76E9Ce4ED33c
公钥
0x037c8cc311d0d8cf36c87e1d13aa02e6acdd807d94eb5
5e40ed64df464bc1db40d
私钥
0xe083e2eecf2e389842d49d5c9d3569ce4d0eb2fe47655b
2124543b0f08c0d005
主网
Rinkeby
Ropsten
Kovan

智能合约是啥?

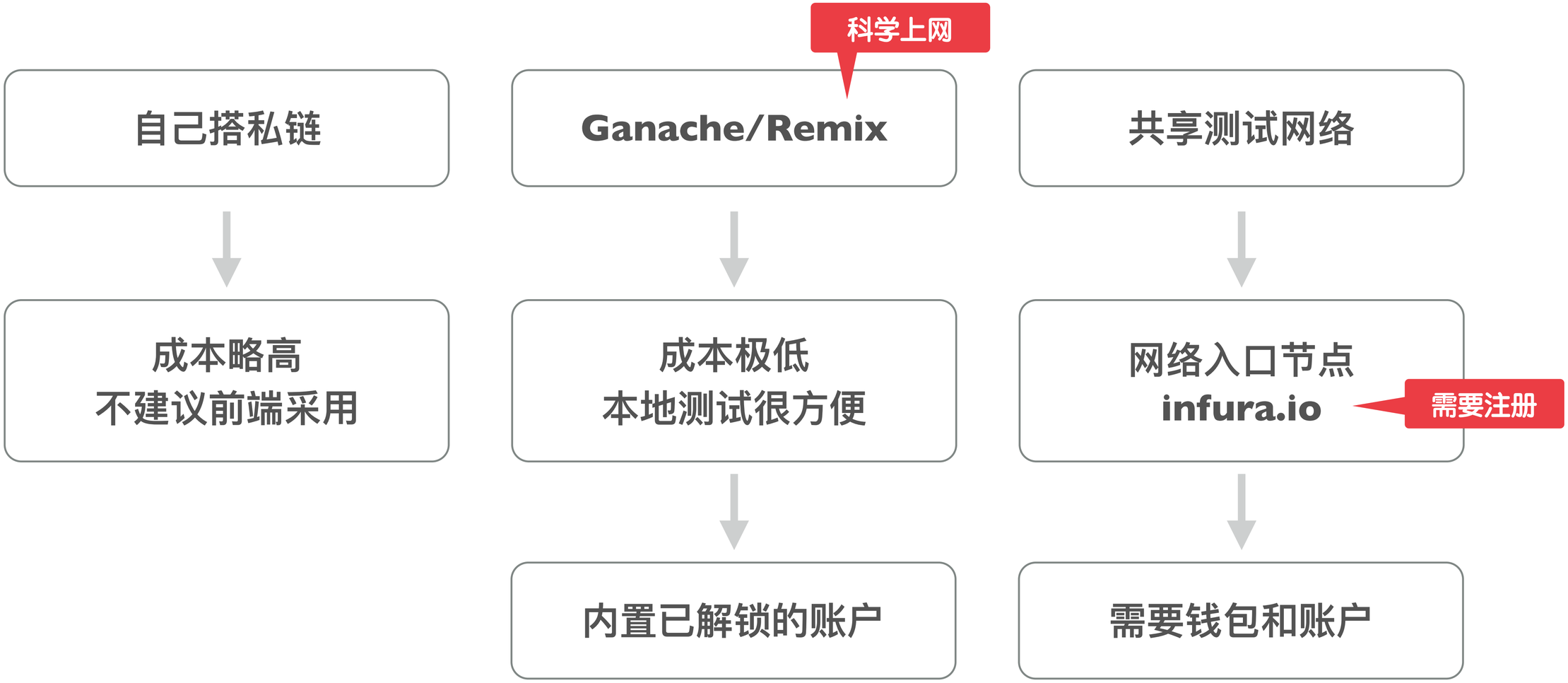

区块链世界
主网：Mainnet
智能合约实例 1
0xd98EFf...5492
智能合约实例 1
0xd98EFf...3628
智能合约实例 1
0xd98EFf...2598
智能合约源代码
测试网络：Rinkeby
智能合约实例 1
0xd98EFf...5492
智能合约实例 1
0xd98EFf...3628
智能合约实例 1
0xd98EFf...2598
...
Solidity
- 强类型
- 面向合约

智能合约和账户的关系


以太坊网络：Rinkeby
智能合约 A
合约间的交互
智能合约 B
合约间交互
智能合约 C
Metamask 账户
imToken 账户
bitpie 账户
用户
用户
用户
通过 Transaction 交互
通过 Transaction 部署
开发者
普通账户 / 外部账户
合约账户 / 内部账户

开发环境准备

必要条件：测试网络


自己搭私链
成本略高
不建议前端采用
科学上网
Ganache/Remix
成本极低
本地测试很方便
内置已解锁的账户
共享测试网络
网络入口节点
infura.io
需要注册
需要钱包和账户

必要条件：账户和余额

交易需要账户才能发起

需要创建钱包和账户

DApp 运行在浏览器中

交易是要收手续费的

浏览器插件

需要注册

使用 Metamask 钱包
metamask.io

使用 Faucet 充值
faucet.rinkeby.io

智能合约工作流

在线 IDE Remix

- 快速验证概念和原型
- 测试已有的合约实例
- 直接部署到以太坊网络

VSCode + Node.js

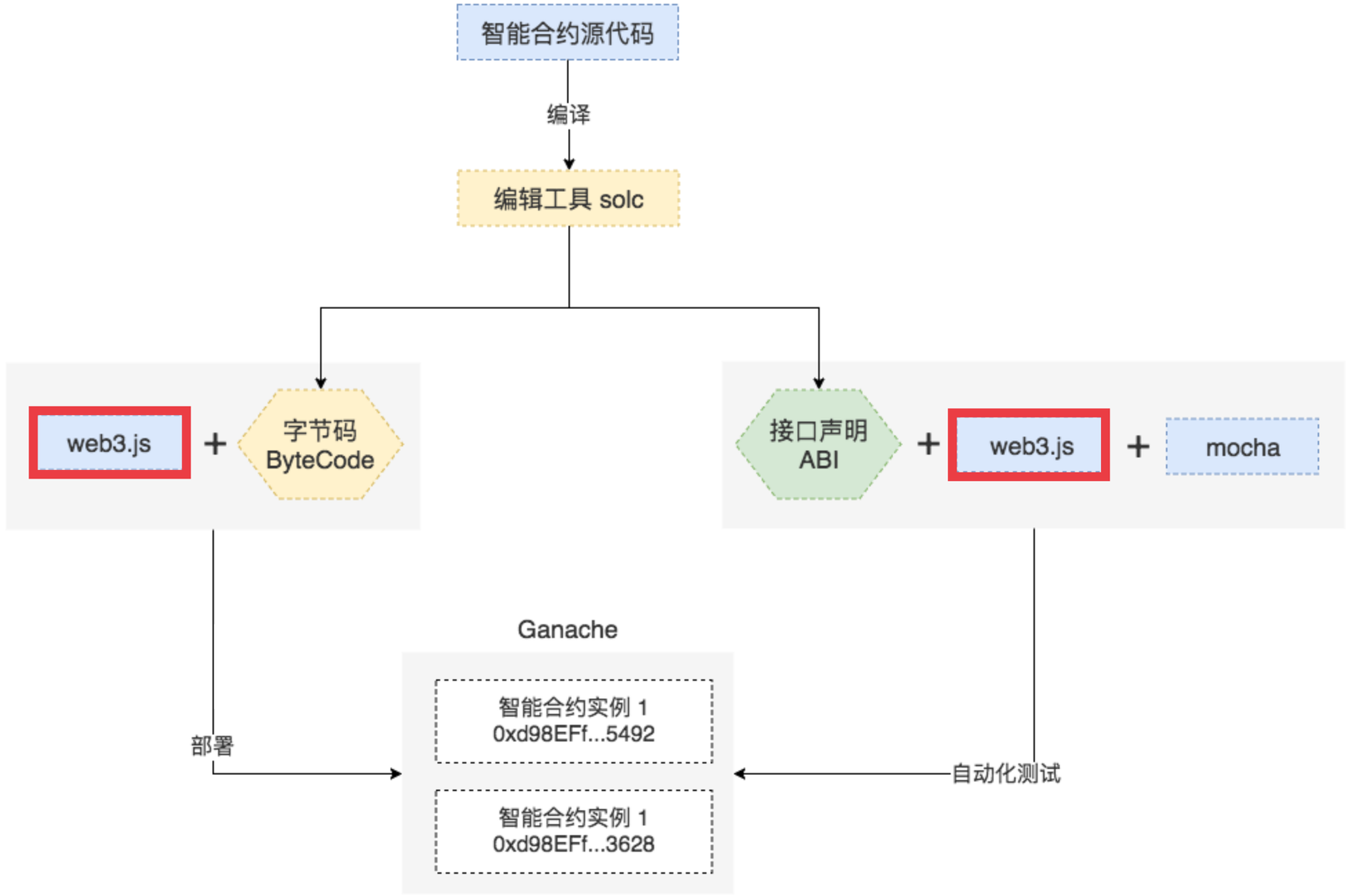
- 可用 Git 来实现版本管理
- 保存编译和部署的结果
- 把能自动化的都自动化

抽奖智能合约

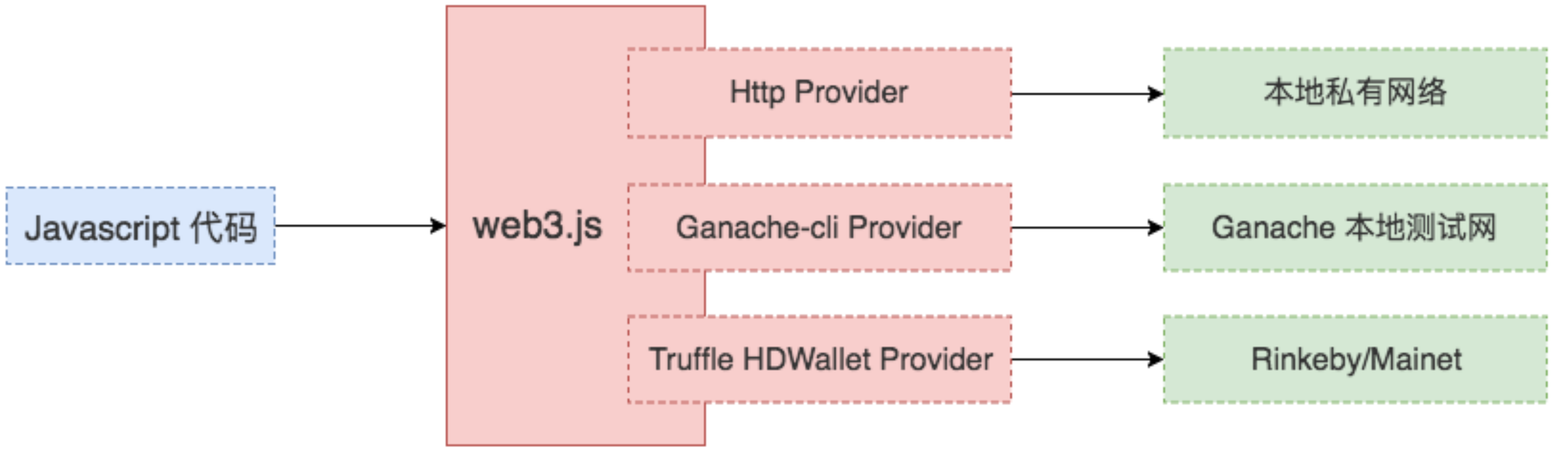
```
pragma solidity ^0.4.17;

contract Lottery {
    address public owner;
    address[] public players;

    constructor() public {
        owner = msg.sender;
    }

    function participate() public payable {
        require(msg.value > .01 ether);
        players.push(msg.sender);
    }

    function random() private view returns(uint) {
        return uint(keccak256(abi.encodePacked(block.difficulty, block.timestamp, players)));
    }

    function pickWinner() public ownerOnly {
        require(players.length > 0);

        uint index = random() % players.length;
        players[index].transfer(address(this).balance);

        players = new address[](0);
    }

    modifier ownerOnly() {
        require(msg.sender == owner);
        _;
    }

    function getPlayers() public view returns(address[]) {
        return players;
    }
}
```

REMIX DEMO

基于 Node.js 的智能合约工作流



web3.js 如何成为桥梁?



智能合约的构建

```
1  const fs = require('fs-extra');
2  const path = require('path');
3  const solc = require('solc');
4
5  const compiledDir = path.resolve(__dirname, '../compiled');
6  fs.removeSync(compiledDir);
7  fs.ensureDirSync(compiledDir);
8
9  const contractFile = 'Lottery.sol';
10 const contractPath = path.resolve(__dirname, '../contracts', contractFile);
11 const contractSource = fs.readFileSync(contractPath, 'utf8');
12 const result = solc.compile(contractSource, 1);
13 console.log(`file compiled: ${contractFile}`);
14
15 console.log(result);
16 if (Array.isArray(result.errors) && result.errors.length) {
17   throw new Error(result.errors[0]);
18 }
19
20 Object.keys(result.contracts).forEach(name => {
21   const contractName = name.replace(/^:/, '');
22   const filePath = path.resolve(compiledDir, `${contractName}.json`);
23   fs.outputJsonSync(filePath, result.contracts[name]);
24   console.log(` > contract ${contractName} saved to ${filePath}`);
25 });
```

准备结果保存目录

编译

错误处理

保存结果

智能合约的测试（上）

```
1  const assert = require('assert');
2  const ganache = require('ganache-cli');
3  const Web3 = require('web3');
4  const { interface, bytecode } = require('../compiled/Lottery.json');
5
6  const web3 = new Web3(ganache.provider());
7
8  let accounts;
9  let contra
10
11 beforeEach(async () => {
12   accounts = await web3.eth.getAccounts();
13
14   contract = await new web3.eth.Contract(JSON.parse(interface))
15     .deploy({ data: bytecode })
16     .send({ from: accounts[0], gas: '1000000' });
17 });
18
19 describe('Lottery Contract', () => {
20   it('should deploy contract', () => {
21     assert.ok(contract.options.address);
22   });
23
24   it('should have owner', async () => {
25     const owner = await contract.methods.owner().call();
26     assert.equal(owner, accounts[0]);
27   });
```

读取编译结果

创建 web3 实例

隔离测试环境

解锁账户

部署合约

测试用例

调用合约接口

智能合约的测试（下）

```
91    it('should send money to winner and reset players array', async () => {
92      await contract.methods.participate().send({
93        from: accounts[0],
94        value: web3.utils.toWei('2', 'ether'),
95      });
96
97      const players = await contract.methods.getPlayers().call({
98        from: accounts[0],
99      });
100     assert.equal(players.length, 1);
101
102     const initialBalance = await web3.eth.getBalance(accounts[0]);
103     await contract.methods.pickWinner().send({ from: accounts[0] });
104     const finalBalance = await web3.eth.getBalance(accounts[0]);
105     const difference = finalBalance - initialBalance;
106
107     console.log({ initialBalance, finalBalance });
108     assert(difference > web3.utils.toWei('1.8', 'ether'));
109
110     const newPlayers = await contract.methods.getPlayers().call({
111       from: accounts[0],
112     });
113     assert.equal(newPlayers.length, 0);
114   });
```

调用合约接口：transaction

调用合约接口：call

检查账户余额

检查合约状态

```
const fs = require('fs-extra');
const path = require('path');
const Web3 = require('web3');
const HDWalletProvider = require('truffle-hdwallet-provider');

const contractPath = path.resolve(__dirname, '../compiled/Lottery.json');
const { interface, bytecode } = require(contractPath);

const provider = new HDWalletProvider(
  'witness improve busy opinion addict sun gossip hedgehog common glass dignity primary',
  'https://rinkeby.infura.io/CqCd0QgCozHBEk19ub2M'
);

const web3 = new Web3(provider);

(async () => {
  const accounts = await web3.eth.getAccounts();
  console.log('合约部署账户:', accounts[0]);

  console.time('合约部署耗时');
  const result = await new web3.eth.Contract(JSON.parse(interface))
    .deploy({ data: bytecode })
    .send({ from: accounts[0], gas: '1000000' });
  console.timeEnd('合约部署耗时');

  const contractAddress = result.options.address;

  console.log('合约部署成功:', contractAddress);
  console.log('合约查看地址:', `https://rinkeby.etherscan.io/address/${contractAddress}`);

  const addressFile = path.resolve(__dirname, '../address.json');
  fs.writeFileSync(addressFile, JSON.stringify(contractAddress));
  console.log('地址写入成功:', addressFile);

  process.exit();
})();
```

钱包助记词

使用 infura 作为入口

部署合约

保存合约地址

把整个流程串起来

用 npm script 打造超溜的前端工作流

DAPP 构建和部署

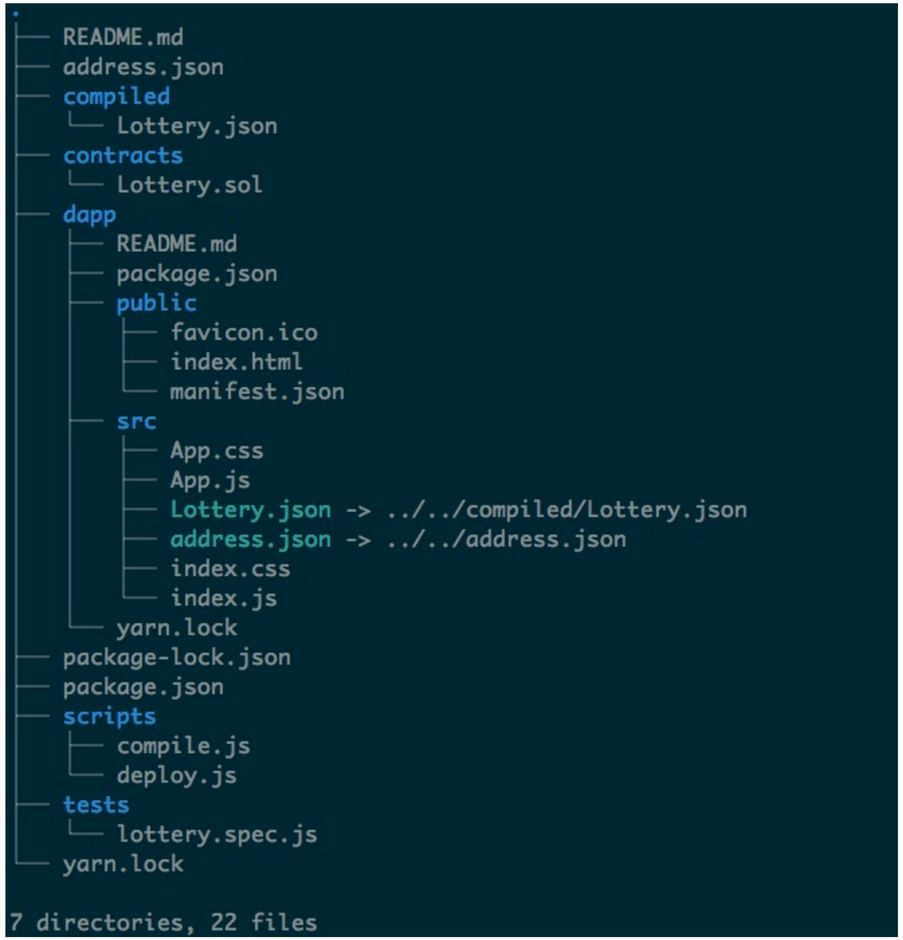
技术视角的 DApp 本质

- 小白用户可以使用的智能合约前端界面
- 能完成智能合约数据读写的前端界面
- 形态可以是 WEB、APP、Desktop
- 简单的 DApp 可做成以区块链为后端的 SPA

前端最擅长的地方

抽奖 DApp 技术选型

- create-react-app
- web3.js



抽奖 DApp 解构


读取合约属性：players
读取合约属性：balance
基于智能合约的抽奖
奖池金额 1 ETH，共 1 人参与抽奖
想试试手气？
输入随机金额
0
参与抽奖
调用合约接口：participate
开奖时间到？
立即开奖
调用合约接口：pickWinner
合约管理员：
0xd98EFfF831aAa4Fe8834F9cb211d8397193A5492
读取合约属性：owner

抽奖 DApp 关键代码：web3 实例化

```
2  import Web3 from 'web3';
3  import address from './address.json';
4  import { interface as ABI } from './Lottery.json';
5
6  import './App.css';
7
8  const web3 = new Web3(window.web3.currentProvider);
9  const contract = new web3.eth.Contract(JSON.parse(ABI), address);
```

Metamask 提供的 provider

抽奖 DApp 关键代码：读取合约数据

```
29    async componentDidMount() {
30      const [owner, players, balance] = await Promise.all([
31        contract.methods.owner().call(),
32        contract.methods.getPlayers().call(),
33        web3.eth.getBalance(contract.options.address),
34      ]);
35
36      console.log({ owner, players, balance });
37
38      this.setState({ owner, players, balance });
39    }
```

所有调用都是异步的

抽奖 DApp 关键代码：渲染合约数据

```
84    render() {
85      const { balance, players, amount, loading, message, owner } = this.state;
86      return (
87        <div className="app">
88          <div className="app-header">
89            <h2 className="app-title">基于智能合约的抽奖</h2>
90          </div>
91          <div className="app-body">
92            <p>
93              奖池金额 {web3.utils.fromWei(balance, 'ether')} ETH, 共 {players.length} 人参与抽奖
94            </p>
95
```

抽奖 DApp 关键代码：修改合约数据

```
45  async participate() {
46    try {
47      this.setState({ message: '参与抽奖中，请稍后 ... ', loading: true });
48
49      const accounts = await web3.eth.getAccounts();
50      await contract.methods.participate().send({
51        from: accounts[0],
52        value: web3.utils.toWei(this.state.amount, 'ether'),
53      });
54
55      this.setState({ message: '参与成功! ', loading: false });
56      setTimeout(() => {
57        window.location.reload();
58      }, 1000);
59    } catch (err) {
60      console.error(err);
61      this.setState({ message: err.message || err.toString(), loading: false });
62    }
63  }
```

交易中状态

注意单位转换

```
104                 <button onClick={this.onParticipate} disabled={loading}>
105                   参与抽奖
106                 </button>
```

抽奖 DAPP DEMO

抽奖 DApp 的部署

- 使用 create-react-app 内置的脚本构建
- 使用 express 来启动极简的 http 服务
- 使用 pm2 管理服务进程
- 使用 npm script 串起来所有步骤

抽奖 DApp 部署：server.js

```
const express = require('express');
const path = require('path');
const app = express();

app.use(express.static(path.join(__dirname, 'dapp/build')));

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'dapp/build', 'index.html'));
});

app.listen(9000);
```

抽奖 DApp 部署：pm2.json

```
{
  "apps": [
    {
      "name": "lottery-dapp",
      "script": "./server.js",
      "out_file": "./logs/out.log",
      "error_file": "./logs/error.log",
      "log_date_format": "YYYY-MM-DD HH:mm:ss",
      "instances": 1,
      "exec_mode": "cluster",
      "max_memory_restart": "500M",
      "merge_logs": true,
      "env": {
        "NODE_ENV": "production"
      }
    }
  ]
}
```

抽奖 DApp 部署： package.json

```
6   "scripts": {
7     "compile": "node scripts/compile.js",
8     "pretest": "npm run compile",
9     "test": "mocha tests/",
10    "predeploy": "npm run test",
11    "deploy": "node scripts/deploy.js",
12    "dev": "cd dapp && npm run start",
13    "build": "cd dapp && npm run build",
14    "prestart": "npm run build",
15    "start": "pm2 restart pm2.json"
16  },
```

构建

部署

抽奖 DAPP 部署演示

想学习更多?

TODO

Q&A